Stacked Capsule Autoencoders

Adam R. Kosiorek[*,†,‡] adamk@robots.ox.ac.uk **Sara Sabour**[§] **Yee Whye Teh**[▽] **Geoffrey E. Hinton**[§]

[‡] **Applied AI Lab** Oxford Robotics Institute University of Oxford

[†] **Department of Statistics** University of Oxford

[§] **Google Brain** Toronto

[▽] **DeepMind** London

Abstract

An object can be seen as a geometrically organized set of interrelated parts. A system that makes explicit use of these geometric relationships to recognize objects should be naturally robust to changes in viewpoint, because the intrinsic geometric relationships are viewpoint-invariant. We describe an unsupervised version of capsule networks, in which a neural encoder, which looks at all of the parts, is used to infer the presence and poses of object capsules. The encoder is trained by backpropagating through a decoder, which predicts the pose of each already discovered part using a mixture of pose predictions. The parts are discovered directly from an image, in a similar manner, by using a neural encoder, which infers parts and their affine transformations. The corresponding decoder models each image pixel as a mixture of predictions made by affine-transformed parts. We learn object- and their part-capsules on unlabeled data, and then cluster the vectors of presences of object capsules. When told the names of these clusters, we achieve state-of-the-art results for unsupervised classification on SVHN (55%) and near state-of-the-art on MNIST (98.5%).

1 Introduction

Convolutional neural networks (CNN) work better than networks without weight-sharing because of their inductive bias: if a local feature is useful in one image location, the same feature is likely to be useful in other locations. It is tempting to exploit other effects of viewpoint changes by replicating features across scale, orientation and other affine degrees of freedom, but this quickly leads to cumbersome high-dimensional feature maps.

An alternative to replicating features across the non-translational degrees of freedom is to explicitly learn transformations between the natural coordinate frame of a whole object and the natural coordinate frames of each of its parts. Computer graphics relies on such object→part coordinate transformations to represent the geometry of an object in a viewpoint-invariant manner. Moreover, there is strong evidence that, unlike standard CNNs, human vision also relies on coordinate frames: imposing an unfamiliar coordinate frame on a familiar object makes it difficult to recognize the object or its geometry (Rock, 1973; Hinton, 1979).

A neural system can learn to reason about transformation between objects, their parts and the viewer, but each of the transformations is likely to require different representation. An object-part-relationship (OP) is viewpoint-invariant and is naturally coded by learned weights. The relationship of an object or part to the viewer changes with the viewpoint (it is viewpoint-equivariant) and is naturally coded using neural activations[2]. With this representation, pose of a single object is represented by its relationship to the viewer. Consequently, representing a single object does not necessitate replicating neural activations across space, unlike in CNNs. It is only processing two (or more) different instances

[*] This work was done during an internship at Google Brain.

[2] This may explain why accessing perceptual knowledge about objects, when they are not visible, requires creating a mental image of the object with a specific viewpoint.

Preprint. Under review.

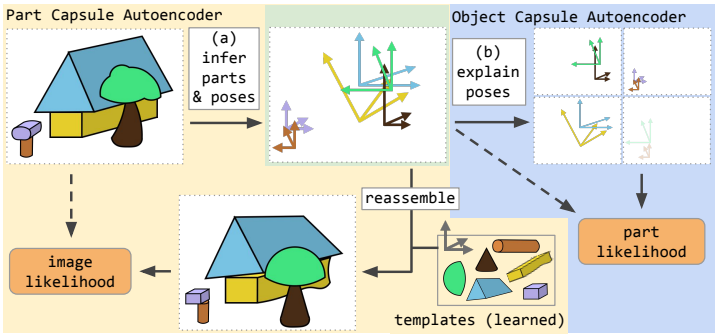

Figure 1: Stacked Capsule Autoencoder (SCAE): (a) *part* capsules segment the input into parts and their poses. The poses are then used to reconstruct the input by affine-transforming learned templates. (b) *object* capsules try to arrange inferred poses into objects, thereby discovering underlying structure. SCAE is trained by maximizing image and part log-likelihoods subject to sparsity constraints.

of the same type of object in parallel that requires spatial replicas of both model parameters and neural activations.

In this paper we propose the Stacked Capsule Autoencoder (SCAE), which has two stages (Fig. 1). The first stage, the Part Capsule Autoencoder (PCAE), segments an image into constituent parts, infers their poses, and reconstructs each image pixel as a mixture of the pixels of transformed part templates. The second stage, the Object Capsule Autoencoder (OCAE), tries to organize discovered parts and their poses into a smaller set of objects that can explain the part poses using a separate mixture of predictions for each part. Every object capsule contributes components to each of these mixtures by multiplying its pose—the object-viewer-relationship (OV)—by the relevant object-part-relationship (OP)[3].

Stacked Capsule Autoencoders (Section 2) capture spatial relationships between whole objects and their parts when trained on unlabelled data. The vectors of presence probabilities for the object capsules tend to form tight clusters, and when we assign a class to each cluster we achieve state-of-the-art results for unsupervised classification on SVHN (55%) and near state-of-the-art on MNIST (98.5%), which can be further improved to 67% and 99%, respectively, by learning fewer than 300 parameters. We also present promising proof-of-concept results on CIFAR10 (Section 3). We describe related work in Section 4 and discuss implications of our work and future directions in Section 5.

2 Stacked Capsule Autoencoders (SCAE)

Segmenting an image into parts is non-trivial, so we begin by abstracting away pixels and the part-discovery stage, and develop the Constellation Capsule Autoencoder (CCAE) (Section 2.1). It uses two-dimensional points as parts, and their coordinates are given as the input to the system. CCAE learns to model sets of points as arrangements of familiar constellations, each of which has been transformed by an independent similarity transform. The CCAE learns to assign individual points to their respective constellations—without knowing the number of constellations or their individual shapes in advance. Next, in Section 2.2, we develop the Part Capsule Autoencoder (PCAE) which learns to infer parts and their poses from images. Finally, we stack the Object Capsule Autoencoder (OCAE), which closely resembles the CCAE, on top of the PCAE to form the Stacked Capsule Autoencoder (SCAE).

2.1 Constellation Autoencoder (CCAE)

Let $\{\mathbf{x}_m \mid m = 1, \ldots, M\}$ be a set of two-dimensional input points, where every point belongs to a constellation as in Figure 2. We first encode all input points (which take the role of part capsules) with Set Transformer (Lee et al., 2019)—a permutation-invariant encoder h^{caps} based on attention mechanisms—into K object capsules. An object capsule k consists of a capsule feature vector $\mathbf{c}_k$, its presence probability $a_k \in [0, 1]$ and a 3×3 object-viewer-relationship (OV) matrix, which represents the affine transformation between the object (constellation) and the viewer. Note that each object capsule can represent only one object at a time. Every object capsule uses a separate multilayer perceptron (MLP) $\text{h}_\text{k}^{\text{part}}$ to predict $N \leq M$ part candidates from the capsule feature vector $\mathbf{c}_k$. Each candidate consists of the conditional probability $a_{k,n} \in [0, 1]$ that a given candidate part exists, an associated scalar standard deviation $\lambda_{k,n}$, and a 3×3 object-part-relationship (OP) matrix, which

[3]The type of a part capsule may determine which, if any, of an object's parts contribute to the mixture used to model the pose of an already discovered part

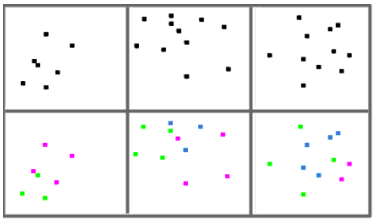

Figure 2: Unsupervised segmentation of points belonging to up to three constellations of squares and triangles at different positions, scales and orientations. The model is trained to reconstruct the points (top row) under the CCAE mixture model. The bottom row colors the points based on the parent with highest posterior probability in the mixture model. The right-most column shows a failure case. Note that the model uses sets of points, not pixels, as its input; we use images only to visualize the constellation arrangements.

represents the affine transformation between the object capsule and the candidate part[4]. Candidate predictions $\mu_{k,n}$ are given by the product of the object capsule OV and the candidate OP matrices. We then model each input part as a Gaussian mixture, where $\mu_{k,n}$ and $\lambda_{k,n}$ are the centers and standard deviations of the isotropic components. See Figures 1 and 5 for illustration; formal description follows:

$$\mathrm{OV}_{1:K}, \mathbf{c}_{1:K}, a_{1:K} = \mathrm{h}^{\mathrm{caps}}(\mathbf{x}_{1:M}) \qquad \text{encode object capsule parameters,} \tag{1}$$

$$\mathrm{OP}_{k,1:N}, a_{k,1:N}, \lambda_{k,1:N} = \mathrm{h}_{\mathrm{k}}^{\mathrm{part}}(\mathbf{c}_k) \qquad \text{decode candidate parameters from } c_k\text{'s,} \tag{2}$$

$$V_{k,n} = \mathrm{OV}_k \mathrm{OP}_{k,n} \qquad \text{decode a part pose candidate,} \tag{3}$$

$$p(\mathbf{x}_m \mid k, n) = \mathcal{N}(\mathbf{x}_m \mid \mu_{k,n}, \lambda_{k,n}) \qquad \text{turn candidates into mixture components,} \tag{4}$$

$$p(\mathbf{x}_{1:M}) = \prod_{m=1}^{M} \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{a_k a_{k,n}}{\sum_i a_i \sum_j a_{i,j}} p(\mathbf{x}_m \mid k, n)\,. \tag{5}$$

The model is trained without supervision by maximizing the likelihood of part capsules in Equation (5) subject to sparsity constraints, *cf.* Section 2.4. The part capsule m can be assigned to the object capsule $k^\star$ as $k^\star = \arg\max_k \; a_k a_{k,n}\, p(\mathbf{x}_m \mid k, n)$.[5] Empirical results show that this model is able to perform unsupervised instance-level segmentation of points belonging to different constellations, even in data which is difficult to interpret for humans. See Figure 2 for an example and Section 3.1 for details.

2.2 Part Capsule Autoencoder (PCAE)

Explaining images as geometrical arrangements of parts requires first inferring what parts the images are composed of, as well as the relationships of the parts to the viewer (which we call their poses). For the CCAE a part is just a 2D point, but here each part capsule has a six degree of freedom (DOF) pose, a presence variable and a unique identity. We frame the part-discovery problem as auto-encoding: the encoder learns to infer the poses and presences of different part capsules, while the decoder learns an image template for each part (Fig. 3) similar to Tieleman, 2014; Eslami et al., 2016. The templates corresponding to present parts are affine-transformed using their poses, and the pixels of these transformed templates are used to create a separate mixture model for each image pixel. The PCAE is followed by an Object Capsule Autoencoder (OCAE), which closely resambles the CCAE and is described in Section 2.3.

Let $\mathbf{y} \in [0,1]^{h \times w \times c}$ be the image. We limit the maximum number of part capsules to M and use an encoder to infer their poses $\mathbf{x}_m \in \mathbb{R}^6$, presence probabilities $d_m \in [0,1]$, and special features $\mathbf{z}_m \in \mathbb{R}^{c_z}$, one per part capsule. The latter do not take part in direct image reconstruction, but inform the OCAE about special aspects of the corresponding part; they are trained by backpropagating derivatives from the OCAE.

At present, we do not allow multiple occurrences of the same type of part in an image, so the part capsules themselves are not replicated across space, though they could be. However, we do need to recognize the part wherever it occurs in the image, and therefore the encoder consists of a CNN with a bottom-up attention mechanism; for every part capsule k, it predicts a feature map $\mathbf{e_k}$ of 6 (pose) + 1 (presence) + c_z (special features) capsule parameters with spatial dimensions $h_e \times w_e$, as well as a single-channel attention mask $\mathbf{a_k}$. The final parameters for that capsule are computed as $\sum_i \sum_j \mathbf{e_{k,i,j}}\, \mathrm{softmax}(\mathbf{a})_{\mathbf{k,i,j}}$, where softmax is along the spatial dimensions. This is similar to global average pooling, but allows some spatial locations to contribute to the final result more than

[4]Deriving these matrices from the capsule feature vector allows for deformable objects. We model OPs as the sum of an input-dependent component and a constant bias. We encourage different capsules to specialize to different constellations by putting a strong L_2 penalty on the former.

[5]We treat parts as independent and evaluate their probability under the same mixture model. While there are no clear 1:1 connections between parts and predictions, it seems to work well in practice.

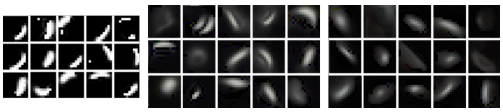

Figure 3: Templates learned on MNIST (left) as well as sobel-filtered SVHN (middle) and CIFAR10 (right). In each case templates converge to strokes. For SVHN they often take the form of double strokes—this is due to sobel filtering, which effectively extracts edges.

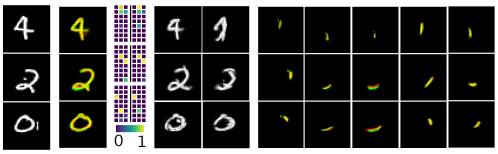

Figure 4: 40×40 MNIST (a) images and their (b) reconstructions from part capsules in red and object capsules in green, with overlapping regions in yellow. Only a few object capsules are activated for every input (c) a priori (left) and even fewer are needed to reconstruct it (right). The most active capsules (d) capture object identity and the majority of information about its appearance. Finally, (e) affine-transformed templates show how exactly parts are used to reconstruct the images.

others; we call this approach *attention-based pooling*. Its effect on the model performance is analyzed in Section 3.3.

The image pixels are modelled as independent Gaussian mixtures. For every pixel, we take the corresponding pixels of the transformed templates and treat them as centers of isotropic Gaussian components with constant variance. Their mixing probabilities are proportional to both presence probabilities of part capsules and a function $f_c : \mathbb{R}^c \mapsto [0, 1]$ of the color value at that location[6], where c is the number of image channels. More formally:

$$\mathbf{x}_{1:M}, d_{1:M}, \mathbf{z}_{1:M} = \mathrm{h}^{\mathrm{enc}}(\mathbf{y}) \qquad \text{encode the image to part capsule parameters,} \quad (6)$$

$$\widehat{T}_m = \mathrm{TransformImage}(T_m, \mathbf{x}_m) \qquad \text{apply affine transforms to image templates,} \quad (7)$$

$$p^y_{m,i,j} \propto d_m \, \mathrm{f_c}\left(\widehat{T}_{m,i,j}\right) \qquad \text{compute mixing probabilities,} \quad (8)$$

$$p(\mathbf{y}) = \prod_{i,j} \sum_{m=1}^{M} p^y_{m,i,j} \, \mathcal{N}\left(y_{i,j} \mid \widehat{T}_{m,i,j}, \sigma_y^2\right) \qquad \text{calculate image likelihood.} \quad (9)$$

2.3 Object Capsule Autoencoder (OCAE)

The next step is to find objects in the already discovered parts[7]. To do so, we use concatenated poses $\mathbf{x}_m$, special features $\mathbf{z}_m$ and flattened templates T_m (which convey the identity of the part capsule) as an input to the OCAE, which differs from the CCAE in the following ways. Firstly, we feed part capsule presence probabilities d_m into the OCAE's encoder—these are used to bias the Set Transformer's attention mechanism to not take absent points into account. Secondly, d_m's are also used to weigh the part-capsules' log-likelihood, *cf.* Equation (5). Additionally, we stop gradient on all of OCAE's inputs except the special features to improve training stability and avoid the problem of collapsing latent variables; see e. g., Rasmus et al., 2015. Finally, parts discovered by the PCAE have independent identities (templates and special features rather than 2D points). Therefore, every part-pose is explained as an independent mixture of predictions from object-capsules—where every object capsule makes exactly M candidate predictions $V_{k,1:M}$, or exactly **one** candidate prediction per part. Consequently, the part-capsule likelihood is given by,

$$p(\mathbf{x}_{1:M}, d_{1:M}) = \prod_{m=1}^{M} \left[\sum_{k=1}^{K} \frac{a_k a_{k,m}}{\sum_i a_i \sum_j a_{i,j}} \, p(\mathbf{x}_m \mid k, m) \right]^{d_m} . \quad (10)$$

2.4 Achieving Sparse and Diverse Capsule Presences

Stacked Capsule Autoencoders are trained to maximise pixel and part log-likelihoods ($\mathcal{L}_{\mathrm{ll}} = \log p(\mathbf{y}) + \log p(\mathbf{x}_{1:M})$). If not constrained, however, they tend to either use all of the part and object capsules to explain every data example, or collapse onto using always the same subset of capsules, regardless of the input. We would like the model to use different sets of part-capsules for different input examples and to specialize object-capsules to particular arrangements of parts;

[6] Templates are assumed to be sparse; if there exists a template that has a non-zero value at a given location, then this templates should be used.

[7] Discovered objects are *not* used top-down to refine the presences or poses of the parts during inference. However, the derivatives backpropagated via OCAE refine the lower-level encoder network that infers the parts.

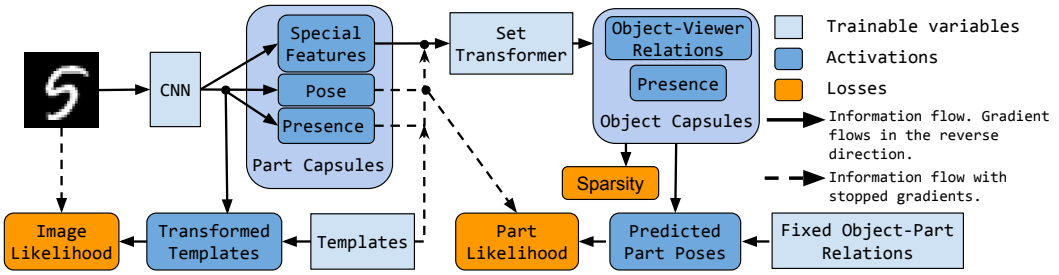

Figure 5: SCAE architecture.

to encourage this, we impose sparsity and entropy constraints. We evaluate their importance in Section 3.3.

We first define prior and posterior object-capsule presence as follows. For a minibatch of size B with K object capsules and M part capsules we define a minibatch of prior capsule presence $a^{\text{prior}}_{1:K}$ with dimension $[B, K]$ and posterior capsule presence $a^{\text{posterior}}_{1:K,1:M}$ with dimension $[B, K, M]$ as,

$$a^{\text{prior}}_{k} = a_k \max_m a_{m,k}\,, \qquad a^{\text{posterior}}_{k,m} = a_k a_{k,m}\,\mathcal{N}(\mathbf{x}_m \mid m, k)\,, \tag{11}$$

respectively; the former is the maximum presence probability among predictions from object capsule k while the latter is the unnormalized mixing probability used to explain part capsule m.

Prior sparsity Let $\overline{u}_k = \frac{1}{B}\sum_{b=1}^{B} a^{\text{prior}}_{b,k}$ the average presence probability of the object capsule k among different training examples, and $\widehat{u}_b = \sum_{k=1}^{K} a^{\text{prior}}_{b,k}$ the sum of object capsule presence probabilities for a given example. If we assume that training examples contain objects from different classes uniformly at random and we would like to assign the same number of object capsules to every class then each class would obtain K/C capsules. Moreover, if we assume that only one object is present in every image, then B/C object capsules should be present for every input example. To this end, we minimize,

$$\mathcal{L}_{\text{prior}} = \frac{1}{B}\sum_{b=1}^{B} ||\widehat{u}_b - \frac{K}{C}||_2 + \frac{1}{K}\sum_{k=1}^{K} ||\overline{u}_k - \frac{B}{C}||_2\,. \tag{12}$$

Posterior Sparsity Similarity, let $\overline{v}_k$ and $\widehat{v}_b$ be the the normalized versions of $\sum_{k,m} a^{\text{posterior}}_{b,k,m}$ and $\sum_{b,m} a^{\text{posterior}}_{b,k,m}$, respectively. We find it beneficial to minimize the within-example entropy of capsule posterior presence $\mathcal{H}(\overline{v}_k)$ and maximize its between-example entropy $\mathcal{H}(\widehat{v}_b)$, where $\mathcal{H}$ is the entropy. The final loss reads as,

$$\mathcal{L}_{\text{posterior}} = \frac{1}{K}\sum_{k=1}^{K} \mathcal{H}(\overline{v}_k) - \frac{1}{B}\sum_{b=1}^{B} \mathcal{H}(\widehat{v}_b)\,. \tag{13}$$

Every active object capsule should explain at least two parts We say that an object capsule has 'won' a part if it has the highest posterior mixing probability for that part among other object capsules. We then create binary labels for each of object capsules, where the label is 1 if the capsule wins at least two parts and it is 0 otherwise. The final loss takes the form of binary cross-entropy between the generated label and the prior capsule presence. This loss is used only for the stand-alone constellation model experiments on point data, *cf.* Sections 2.1 and 3.1.

Fig. 5 shows the schematic architecture of SCAE. We optimize a weighted sum of image and part likelihoods and the auxiliary losses. Loss weight selection process as well as the values used for experiments are explained in Appendix A.

In order to make the values of presence probabilities ($a_k, a_{k,m}$ and d_m) closer to binary we inject uniform noise $\in [-2, 2]$ into logits, similar to Tieleman, 2014. This forces the model to predict logits that are far from zero to avoid stochasticity and makes the predicted presence probabilities close to binary. Interestingly, it tends to work better in our case than using the Concrete distribution (Maddison et al., 2017).

3 Evaluation

The decoders in the SCAE use explicitly parameterised affine transformations that allow the encoders' inputs to be explained with a small set of transformed objects or parts. The following evaluations show

Table 1: Unsupervised classification results in % with (standard deviation) are averaged over 5 runs. Methods based on mutual information are shaded. Results marked with † use data augmentation, ∇ use IMAGENET-pretrained features instead of images, while § are taken from Ji et al., 2018. We highlight the best results and those that are are within its 98% confidence interval according to a two-sided t test.

Method	MNIST	CIFAR10	SVHN
KMEANS (Haeusser et al., 2018)	53.49	20.8	12.5
AE (Bengio et al., 2007)[§]	81.2	31.4	-
GAN (Radford et al., 2016)[§]	82.8	31.5	-
IMSAT (Hu et al., 2017)[†,∇]	**98.4** (0.4)	45.6 (0.8)	**57.3** (3.9)
IIC (Ji et al., 2018)[§,†]	**98.4** (0.6)	**57.6** (5.0)	-
ADC (Haeusser et al., 2018)[†]	**98.7** (0.6)	29.3 (1.5)	38.6 (4.1)
MAX-ACT (SCAE)	**98.0 (.15)**	19.79 (1.0)	49.07 (1.7)
CLUST-NN (SCAE)	**98.5 (.11)**	19.39 (1.5)	**53.0 (3.8)**
LIN-MATCH (SCAE)	**98.5 (.10)**	25.01 (1.0)	**55.33 (3.4)**
LIN-PRED (SCAE)	**98.9 (.07)**	33.48 (0.3)	**67.27 (4.5)**

how the embedded geometrical knowledge helps to discover patterns in data. Firstly, we show that the CCAE discovers underlying structures in arrangements of constellations made of two-dimensional points, thereby performing instance-level segmentation. Secondly, we pair an OCAE with a PCAE and investigate whether the resulting SCAE can discover structure in real images. Finally, we present an ablation study that shows which components of the model contribute to the results.

3.1 Discovering Constellations

We create arrangements of constellations online, where every input example consists of up to 11 two-dimensional points belonging to up to three different constellations (two squares and a triangle) as well as binary variables indicating presence of the points (points can be missing). Each constellation is included with probability 0.5 and undergoes a similarity transformation, whereby it is randomly scaled, rotated by up to 180° and shifted. Finally, every input example is normalized such that all points lie within $[-1, 1]^2$. Note that we use sets of points, and not images, as inputs to our model.

We compare the CCAE against a baseline that uses the same encoder but a simpler decoder: the decoder uses the capsule parameter vector $\mathbf{c}_k$ to directly predict the location, precision and presence probability of each of the four points as well as the presence probability of the whole corresponding constellation. Implementation details are listed in Appendix A.1.

Both models are trained unsupervised by maximizing the part log-likelihood. We evaluate them by trying to assign each input point to one of the object capsules. To do so, we assign every input point to the object capsule with the highest posterior probability for this point, *cf.* Section 2.1, and compute segmentation accuracy (i. e., the true-positive rate).

The CCAE consistently achieves below 4% error with the best model achieving 2.8%, while the best baseline achieved 26% error using the same budget for hyperparameter search. This shows that wiring in an inductive bias towards modelling geometric relationships can help to bring down the error by an order of magnitude—at least in a toy setup where each set of points is composed of familiar constellations that have been independently transformed.

3.2 Unsupervised Class Discovery

To allow for multimodality in the appearance of objects of a specific class, we typically use more object capsules than the number of class labels. We expect that the vector of presence probabilities of object capsules should be highly informative of the class label. To test this hypothesis, we train SCAE on MNIST, SVHN and CIFAR10 and try to assign class labels to vectors of object capsule presences. This is done with one of the following methods: MAX-ACT: we search for a training example that maximally activates given object capsule and assign the corresponding label to this capsule; CLUSTER-NN: we perform KMEANS clustering into C clusters and then find the training example that is the closest to each cluster's centroid to assign a label to the cluster; LIN-MATCH: after finding 10 clusters[8] with KMEANS we use bipartite graph matching (Kuhn, 1955) to find the permutation of cluster indices that minimizes the classification error—this is standard practice in unsupervised classification, see e. g., Ji et al., 2018; LIN-PRED: we train a linear classifier with supervision given the presence vectors; this learns $K \times 10$ weights and 10 biases, where K is the number of object capsules, but it does not modify any parameters of the main model.

[8] All considered datasets have 10 classes.

Table 2: Ablation study on MNIST. All used model components contribute to its final performance. AFFNIST results show out-of-distribution generalization properties and come from a model trained on 40×40 MNIST. Numbers represent average % and (standard deviation) over 10 runs. We highlight the best results and those that are are within its 98% confidence interval according to a two-sided t test.

	Method	MNIST	40×40 MNIST	AFFNIST
	full model	**97.0 (.87)**	**98.5 (.1)**	**92.2 (.59)**
a)	no posterior sparsity	**96.7 (.7)**	**98.2 (.48)**	87.6 (1.63)
	no prior sparsity	90.5 (7.56)	94.0 (3.03)	74.0 (4.94)
	no prior/posterior sparsity	63.0 (13.48)	62.7 (10.46)	40.7 (6.81)
b)	no noise in object caps	**96.4 (1.41)**	**97.8 (.67)**	**90.8 (2.97)**
	no noise in any caps	84.8 (6.22)	85.1 (13.13)	76.3 (12.89)
	no noise in part caps	83.9 (7.57)	80.2 (9.1)	73 (9.04)
c)	similarity transforms	90.4 (13.78)	**97.4 (.99)**	**90.1 (2.62)**
	no deformations	87.6 (6.13)	95.2 (1.04)	87.6 (1.26)
d)	LINEAR part enc	**94.8 (3.0)**	98.1 (.26)	76.3 (2.22)
	CONV part enc	**96.3 (.85)**	**97.8 (.95)**	80.1 (2.58)
e)	MLP enc for object caps	73.0 (6.34)	70.3 (11.2)	52.5 (11.29)
f)	no special features	63.1 (10.55)	66.9 (23.59)	50.5 (18.26)

In agreement with previous work on unsupervised clustering (Ji et al., 2018; Hu et al., 2017; Hjelm et al., 2019; Haeusser et al., 2018), we train our models and report results on full datasets (TRAIN, VALID and TEST splits). The linear transformation used in LIN-PRED variant of our method is trained on the TRAIN split of respective datasets while its performance on the TEST split is reported.

We used an PCAE with 24 single-channel 11×11 templates for MNIST and 24 and 32 three-channel 14×14 templates for SVHN and CIFAR10, respectively. We used sobel-filtered images as the reconstruction target for SVHN and CIFAR10, as in Jaiswal et al., 2018, while using the raw pixel intensities as the input to PCAE. The OCAE used 24, 32 and 64 object capsules, respectively. Further details on model architectures and hyper-parameter tuning are available in Appendix A. All results are presented in Table 1. SCAE achieves competitive results in unsupervised object classification on MNIST and SVHN and under-performs slightly on CIFAR10, which is further discussed in Section 5.

3.3 Ablation study

SCAEs have many moving parts; an ablation study shows which model components are important and to what degree. We train SCAE variants on MNIST as well as a padded-and-translated 40×40 version of the dataset, where the original digits are translated up to 6 pixels in each direction. Trained models are tested on TEST splits of both datasets; additionally, we evaluate the model trained on the 40×40 MNIST on the TEST split of AFFNIST dataset. Testing on AFFNIST shows whether the model can generalize to unseen viewpoints. This task was used by Rawlinson et al., 2018 to evaluate Sparse Unsupervised Capsules, which achieved 90.12% accuracy. SCAE achieves $92.2 \pm 0.59\%$, which indicates that it is better at viewpoint generalization. We choose the LIN-MATCH performance metric, since it is the one favoured by the unsupervised classification community. Results are split into several groups and shown in Table 2. We describe each group in turn. Group a) shows that sparsity losses introduced in Section 2.4 increase model performance, but that the posterior loss might not be necessary. Group b) checks the influence of injecting noise into logits for presence probabilities, *cf.* Section 2.4. Injecting noise into part capsules seems critical, while noise in object capsules seems unnecessary—the latter might be due to sparsity losses. Group c) shows that using similarity (as opposed to affine) transforms in the decoder can be restrictive in some cases, while not allowing deformations hurts performance in every case.

Group d) evaluates the type of the part-capsule encoder. The LINEAR encoder entails a CNN followed by a fully-connected layer, while the CONV encoder predicts one feature map for every capsule parameter, followed by global-average pooling. The choice of part-capsule encoder seems not to matter much for within-distribution performance; however, our attention-based pooling does achieve much higher classification accuracy when evaluated on a different dataset, showing better generalization to novel viewpoints.

Additionally, e) using Set Transformer as the object-capsule encoder is essential. We hypothesize that it is due to the natural tendency of Set Transformer to find clusters, as reported in Lee et al., 2019. Finally, f) using special features $\mathbf{z}_m$ seems not less important—presumably due to effects the high-level capsules have on the representation learned by the primary encoder.

4 Related Work

Capsule Networks Our work combines ideas from Transforming Autoencoders (Hinton, Krizhevsky, et al., 2011) and EM Capsules (Hinton, Sabour, et al., 2018). Transforming autoencoders discover affine-aware capsule *instantiation parameters* by training an autoencoder to predict an affine-transformed version of the input image from the original image plus an extra input, which explicitly represents the transformation. By contrast, our model does not need any input other than the image.

Both EM Capsules and the preceding Dynamic Capsules (Sabour et al., 2017) use the poses of parts and learned part→object relationships to vote for the poses of objects. When multiple parts cast very similar votes, the object is assumed to be present, which is facilitated by an interactive inference (routing) algorithm. Iterative routing is inefficient and has prompted further research. Wang and Liu, 2018 formulated routing as an optimization of a clustering loss and a KL-divergence-based regularization term. Zhang et al., 2018 proposed a weighted kernel density estimation-based routing method. Li et al., 2018 proposed approximating routing with two branches and sending feedback via optimal transport divergence between two distributions (lower and higher capsules). In contrast to prior work, we use objects to predicts parts rather than vice-versa, therefore we can dispense with iterative routing at inference time. The encoder of the OCAE learns how to group parts into objects and it respects the single parent constraint, because it is trained using derivatives produced by a decoder that uses a mixture model of parts which assumes that each part must be explained by a single object.

Additionally, since it is the objects that predict parts, the parts are allowed to have fewer degrees-of-freedom in their poses than objects (as in the CCAE). Inference is still possible, because the OCAE encoder makes object predictions based on *all* the parts rather than an individual part.

A further advantage of our version of capsules is that it can perform unsupervised learning. Previous versions of capsules used discriminative learning, though Rawlinson et al., 2018 used the reconstruction MLP introduced in Sabour et al., 2017 to train Dynamic Capsules without supervision and has shown that unsupervised training for capsule-conditioned reconstruction helps with generalization to AFFNIST classification; we further improve on their results, *cf.* Section 3.3.

Unsupervised Classification There are two main approaches to unsupervised object category detection in computer vision. The first one is based on representation learning and typically requires discovering clusters or learning a classifier on top of the learned representation. Eslami et al., 2016; Kosiorek et al., 2018 use an iterative procedure to infer a variable number of latent variables, one for every object in a scene, that are highly informative of object class, while Greff et al., 2019; Burgess et al., 2019 perform unsupervised instance-level segmentation in an iterative fashion. While similar to our work, these approaches cannot decompose objects into their constituent parts and do not provide explicit description of object shape (e. g., templates and their poses in our model).

The second approach targets classification explicitly by minimizing mutual information (MI)-based losses and directly learning class-assignment probabilities. IIC (Ji et al., 2018) maximizes an exact estimator of MI between two discrete probability vectors describing (transformed) versions of the input image. DeepInfoMax (Hjelm et al., 2019) relies on negative samples and maximizes MI between the predicted probability vector and its input via noise-contrastive estimation (Gutmann and Hyvärinen, 2010). This class of methods directly maximizes the amount of information contained in an assignment to discrete clusters and they hold state-of-the-art results on most unsupervised classification tasks. MI-based methods suffer from typical drawbacks of mutual information estimation: they require heavy data augmentation and large batch sizes. This is in contrast to our method, which achieves comparable performance with batch size no bigger than 128 and with no data augmentation.

Geometrical Reasoning Other attempts at incorporating geometrical knowledge into neural networks include exploiting equivariance properties of group transformations (Cohen and Welling, 2016) or new types of convolutional filters (Oyallon and Mallat, 2015; Dieleman et al., 2016). Although they achieve significant parameter efficiency in handling rotations or reflections compared to standard CNNs, these methods cannot handle additional degrees of freedom of affine transformations—like scale. Lenssen et al., 2018 combined capsule networks with group convolutions to guarantee equivariance and invariance in capsule networks. Spatial Transformers (ST; Jaderberg et al., 2015) apply affine transformations to the image sampling grid while steerable networks (Cohen and Welling, 2017; Jacobsen et al., 2017) dynamically change convolutional filters. These methods are similar to ours in the sense that transformation parameters are predicted by a neural network, but differ in the sense that ST uses global transformations applied to the whole image while steerable networks use

only local transformations. Our approach can use different global transformations for every object as well as local transformations for each of their parts.

5 Discussion

The main contribution of our work is a novel method for representation learning, in which highly structured decoder networks are used to train one encoder network that can segment an image into parts and their poses and another encoder network that can compose the parts into coherent wholes. Despite the fact that our training objective is not concerned with classification or clustering, SCAE is the only method that achieves competitive results in unsupervised object classification without relying on mutual information (MI). This is significant, since unlike our method, MI-based methods require sophisticated data augmentation. It may be possible to further improve results by using an MI-based loss to train SCAE, where the vector of capsule probabilities could take the role of discrete probability vectors in IIC (Ji et al., 2018). SCAE under-performs on CIFAR10, which could be because of using fixed templates, which are not expressive enough to model real data. This might be fixed by building deeper hierarchies of capsule autoencoders (e. g., complicated scenes in computer graphics are modelled as deep trees of affine-transformed geometric primitives) as well as using input-dependent shape functions instead of fixed templates—both of which are promising directions for future work. It may also be possible to make a much better PCAE for learning the primary capsules by using a differentiable renderer in the generative model that reconstructs pixels from the primary capsules.

Finally, the SCAE could be the 'figure' component of a mixture model that also includes a versatile 'ground' component that can be used to account for everything except the figure. A complex image could then be analyzed using sequential attention to perceive one figure at a time.

6 Acknowledgements

We would like to thank Sandy H. Huang for help with editing the manuscript and making Figure 1. Additionally, we would like to thank S. M. Ali Eslami and Danijar Hafner for helpful discussions throughout the project. We also thank Hyunjik Kim, Martin Engelcke, Emilien Dupont and Simon Kornblith for feedback on initial versions of the manuscript.

References

J. Ba, J. Kiros, and G. E. Hinton (2016). "Layer Normalization". In: *CoRR*. arXiv: 1607.06450.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle (2007). "Greedy Layer-wise Training of Deep Networks". In: *Advances in Neural Information Processing Systems*.

C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner (2019). "MONet: Unsupervised Scene Decomposition and Representation". In: *CoRR*. arXiv: 1901.11390.

T. Cohen and M. Welling (2016). "Group Equivariant Convolutional Networks". In: *International Conference on Machine Learning*.

T. Cohen and M. Welling (2017). "Steerable CNNs". In: *International Conference on Representation Learning*.

S. Dieleman, J. De Fauw, and K. Kavukcuoglu (2016). "Exploiting Cyclic Symmetry in Convolutional Neural Networks". In: *CoRR*. arXiv: 1602.02660.

S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K. Kavukcuoglu, and G. E. Hinton (2016). "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models". In: *Advances in Neural Information Processing Systems*. arXiv: 1603.08575.

K. Greff, R. L. Kaufmann, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner (2019). "Multi-Object Representation Learning with Iterative Variational Inference". In: *arXiv preprint arXiv:1903.00450*.

M. Gutmann and A. Hyvärinen (2010). "Noise-contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models". In: *International Conference on Artificial Intelligence and Statistics*.

P. Haeusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers (2018). "Associative Deep Clustering: Training a Classification Network with No Labels". In: *German Conference on Pattern Recognition*.

G. E. Hinton (1979). "Some Demonstrations of the Effects of Structural Descriptions in Mental Imagery". In: *Cognitive Science* 3.

G. E. Hinton, A. Krizhevsky, and S. D. Wang (2011). "Transforming Auto-Encoders". In: *International Conference on Artifical Neural Networks*.

G. E. Hinton, S. Sabour, and N. Frosst (2018). "Matrix Capsules with EM routing". In: *International Conference on Learning Representations*.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio (2019). "Learning Deep Representations by Mutual Information Estimation and Maximization". In: *CoRR*. arXiv: 1808.06670.

W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama (2017). "Learning Discrete Representations via Information Maximizing Self-augmented Training". In: *International Conference on Machine Learning*.

J.-H. Jacobsen, B. De Brabandere, and A. W. Smeulders (2017). "Dynamic steerable blocks in deep residual networks". In: *CoRR*. arXiv: 1706.00598.

M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu (2015). "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems*. DOI: 10.1038/nbt.3343. arXiv: 1506.02025v1.

A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan (2018). "Capsulegan: Generative adversarial capsule network". In: *European Conference on Computer Vision (ECCV)*.

X. Ji, J. F. Henriques, and A. Vedaldi (2018). "Invariant Information Distillation for Unsupervised Image Segmentation and Clustering". In: *CoRR*. arXiv: 1807.06653. URL: http://arxiv.org/abs/1807.06653.

A. Kosiorek, H. Kim, Y. W. Teh, and I. Posner (2018). "Sequential Attend, Infer, Repeat: Generative modelling of moving objects". In: *Advances in Neural Information Processing Systems*. arXiv: 1806.01794.

H. W. Kuhn (1955). "The Hungarian Method for the Assignment Problem". In: *Naval Research Logistics Quarterly*.

J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh (2019). "Set Transformer". In: *International Conference on Machine Learning*. arXiv: 1810.00825.

J. E. Lenssen, M. Fey, and P. Libuschewski (2018). "Group Equivariant Capsule Networks". In: *Advances in Neural Information Processing Systems*.

H. Li, X. Guo, B. Dai, W. Ouyang, and X. Wang (2018). "Neural Network Encapsulation". In: *CoRR*. arXiv: 1808.03749.

C. J. Maddison, A. Mnih, and Y. W. Teh (2017). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *International Conference on Learning Representations*.

E. Oyallon and S. Mallat (2015). "Deep Roto-Translation Scattering for Object Classification". In: *IEEE Conference on Computer Vision and Pattern Recognition*.

A. Radford, L. Metz, and S. Chintala (2016). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *International Conference on Learning Representations*.

A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko (2015). "Semi-supervised Learning with Ladder Networks". In: *Advances in Neural Information Processing Systems*.

D. Rawlinson, A. Ahmed, and G. Kowadlo (2018). "Sparse Unsupervised Capsules Generalize Better". In: *CoRR*. arXiv: 1804.06094.

I. Rock (1973). *Orientation and form*. Academic Press.

S. Sabour, N. Frosst, and G. E. Hinton (2017). "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems*.

T. Tieleman and G. E. Hinton (2012). *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning.

T. Tieleman (2014). *Optimizing Neural Networks That Generate Images*. University of Toronto, Canada.

D. Wang and Q. Liu (2018). "An Optimization View on Dynamic Routing Between Capsules". In:

S. Zhang, Q. Zhou, and X. Wu (2018). "Fast Dynamic Routing Based on Weighted Kernel Density Estimation". In: *International Symposium on Artificial Intelligence and Robotics*.

A Model Details

A.1 Constellation Experiments

The CCAE uses a four-layer Set Transformer as its encoder. Every layer has four attention heads, 128 hidden units per head, and is followed by layer norm (Ba et al., 2016). The encoder outputs three 32-dimensional vectors—one for each object capsule. The decoder uses a separate neural net for each object capsule to predict all parameters used to model its points: this includes four candidate part predictions per capsule for a total of 12 candidates. In this experiment, each object→part relationship OP is just a 2-D offset in the object's frame of reference (instead of a 3×3 matrix) and it is affine transformed by the corresponding OV matrix to predict the 2-D point.

A.2 Image Experiments

We use a convolutional encoder for part capsules and a set transformer encoder (Lee et al., 2019) for object capsules. Decoding from object capsule to part capsules is done with MLPs, while the input image is reconstructed with affine-transformed learned templates. Details of the architectures we used are available in Table 3.

Table 3: Architecture details. S in the last column means that the entry is the same as for SVHN.

Dataset	Constellation	MNIST	SVHN	CIFAR10
num templates	N/A	24	24	32
template size	N/A	11×11	14×14	S
num capsules	3	24	32	64
part CNN	N/A	2x(128:2)-2x(128:1)	2x(128:1)-2x(128:2)	S
set transformer	4x(4-128)-32	3x(1-16)-256	3x(2-64)-128	S

We use ReLu nonlinearities except for presence probabilities, for which we use sigmoids. (128:2) for a CNN means 128 channels with a stride of two. All kernels are 3×3. For set transformer (1-16)-256 means one attention head, 16 hidden units and 256 output units; it uses layer normalization (Ba et al., 2016) as in the original paper (Lee et al., 2019) but no dropout. All experiments (apart from constellations) used 16 special features per part capsule.

For SVHN and CIFAR10, we use normalized sobel-filtered images as the target of the reconstruction to emphasize the shape importance. Figure 6 in Appendix B shows examples of SVHN and CIFAR10 reconstruction. The filtering procedure is as follows: 1) apply sobel filtering, 2) subtract the median color, 3) take the absolute value of the image, 4) normalize for image values to be $\in [0, 1]$.

All models are trained with the RMSProp optimizer (Tieleman and Hinton, 2012) $\mathrm{momentum} = .9$ and $\epsilon = (10 * \mathrm{batch\_size})^{-2}$. Batch size is 64 for constellations and 128 for all other datasets. The learning rate was equal to 10^{-5} for MNIST and constellation experiments (without any decay), while we run a hyperparameter search for SVHN and CIFAR10: we searched learning rates in the range of $5 * 10^{-5}$ to $5 * 10^{-4}$ and exponential learning rate decay of 0.96 every 10^5 or $3 * 10^5$ weight updates. Learning rate of 10^{-4} was selected for both SVHN and CIFAR10, the decay steps was 10^5 for SVHN and $3 * 10^5$ for CIFAR10. The LIN-PRED accuracy on a validation set is used as a proxy to select the best hyperparameters—including weights on different losses, reported in Table 4. Models were trained for up to $3 * 10^5$ iterations on single Tesla V100 GPUs, which took 40 minutes for constellation experiments and less than a day for CIFAR10.

Table 4: Loss weights values. The *within* and *between* quantifiers in sparsity losses corresponds to different terms of Equations (12) and (13).

Dataset	Constellation	MNIST	SVHN	CIFAR10
part ll weight	1	1	2.56	2.075
image ll weight	N/A	1	1	1
prior within sparsity	1	1	0.22	0.17
prior between sparsity	1	1	0.1	0.1
posterior within sparsity	0	10	8.62	1.39
posterior between sparsity	0	10	0.26	7.32
too-few-active-capsules	10	0	0	0

B Reconstructions

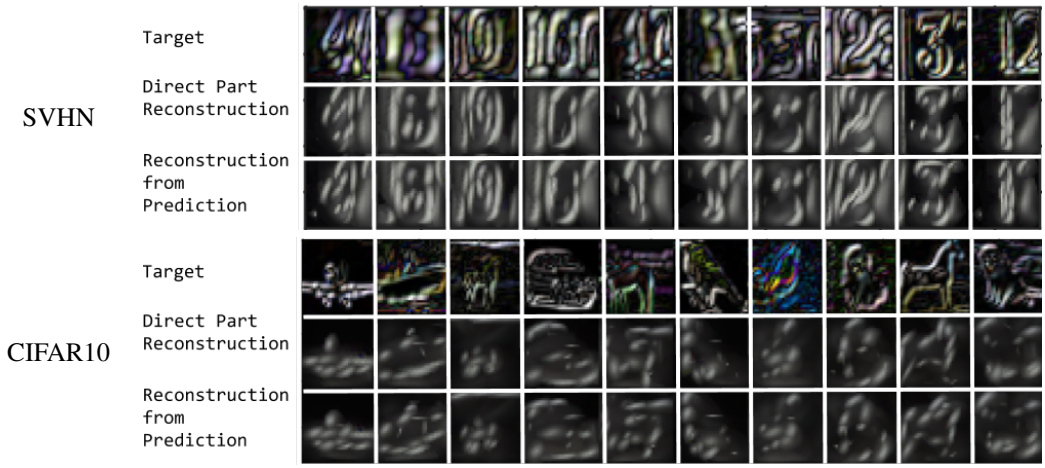

Figure 6: 10 Sample SVHN and Cifar10 reconstructions. First row shows Sobel filtered target image. Second row shows the reconstruction from Part Capsule Layer directly. Third row shows the reconstruction if we use the object predictions for the Part poses instead of Part poses themselves for reconstruction. The templates in this model has the same number of channels as the image, but they have converged to black and white templates and the reconstruction do not have color diversity. The SCAE model is trained completely unsupervised but the reconstructions tend to focus on the center digit in SVHN and filter the rest of the clutter.